DOCUMENT RESUME

ED 327 153 IR 014 777

AUTHOR Milet, Lynn K.; Harvey, Francis A.
TITLE An Exploration and Analysis of the Relationships among Object Oriented Programming, Hypermedia, and Hypertalk.
PUB DATE Nov 89
NOTE 6p.; Paper presented at the International Meeting of the Association for the Development of Computer-Based Instructional Systems (31st, Washington, DC, November 12-16, 1989).
PUB TYPE Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Comparative Analysis; Computer System Design; *Hypermedia; *Information Retrieval; Information Systems; Interaction; *Man Machine Systems; Programing; Users (Information)
IDENTIFIERS *Object Oriented Programing Systems

ABSTRACT

Hypermedia and object oriented programming systems (OOPs) represent examples of "open" computer environments that allow the user access to parts of the code or operating system. Both systems share fundamental intellectual concepts (objects, messages, methods, classes, and inheritance), so that an understanding of hypermedia can help in the understanding of OOPs and vice versa. The degree to which the concepts are intellectually and operationally parallel, however, differs markedly for particular hypermedia applications. Other comparisons reveal that both systems are designed to emulate the way humans store and retrieve data, and both allow a non-linear method of searching for information. Hypermedia and OOPs appear to be the likely tools for the development of interactive user interfaces to other developed programs. They are both systems that force the programmer to look at knowledge from the learner's perspective and needs. (DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

AN EXPLORATION AND ANALYSIS OF THE RELATIONSHIPS AMONG OBJECT ORIENTED PROGRAMMING, HYPERMEDIA, AND HYPERTALK

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Lynn K. Milet
Francis A. Harvey
Lehigh University

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Lynn Milet
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED327153

ABSTRACT

This paper presents an overview of the fundamental concepts of both object oriented programming systems and hypermedia. Hypermedia and object oriented programming systems (OOPs) represent examples of "open" computer environments which allow the user some access to parts of the code or operating system. Both systems share many fundamental intellectual constructs, so that an understanding of hypermedia can help in the understanding of OOPs and vice versa.

The fundamental concepts of OOPs (objects, messages, methods, classes, and inheritance) each have some counterparts in hypermedia systems. The degree to which the counterparts are intellectually and operationally parallel, however, differs markedly for particular hypermedia applications. In HyperTalk (the programming environment for HyperCard applications on Apple Macintosh), for example, cards and buttons are analogous in some ways to objects, and handlers and scripts are somewhat analogous to messages. HyperTalk, however, does not support inheritance. This paper explores and analyzes these and other relationships and shows how an understanding of OOPs can help in understanding hypermedia and in developing hypermedia applications.

INTRODUCTION

In 1945, Vannevar Bush described his vision of the future of information retrieval systems. This dream, called Memex, provided access to online text and graphic information, the ability to enter notes and other information by the user, and the ability to link any two items together. Although Memex was never actually developed, it provided the conceptual basis for more sophisticated interactive information retrieval systems, such as hypermedia and object-oriented programming systems.

Large software systems are often complicated and easily disturbed by outside influences. Some systems choose a "closed" approach to change by not allowing the user entry to either the code or the operating system. (Examples are commercial spreadsheets like Lotus 123; and word processors like WordStar.). Other programs are developed using an "open" policy in which the user has access to the code, the operating system, or both. Hypertext and object-oriented programming systems are two such "open" environments currently under development for information retrieval applications.

Hypermedia, also referred to as interactive multimedia, holds great potential for education and training. Object-oriented programming offers a powerful conceptual tool for analyzing and solving computer-based problems, particularly problems utilizing advanced technology applications. This paper will present an overview of the fundamental concepts of both object-oriented programming systems and hypermedia and a comparison of the two systems.

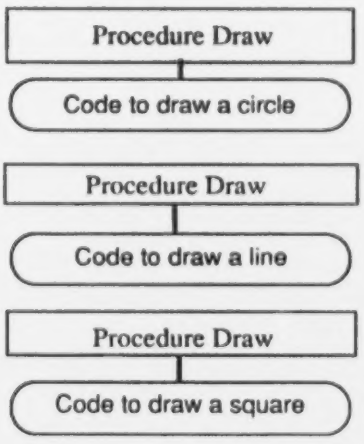
OBJECT-ORIENTED PROGRAMMING

Object-oriented programming (OOP) languages have been used to develop database

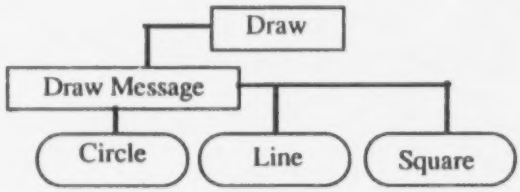
systems, authoring systems, and expert systems. Some examples of OOP languages used are Smalltalk, C++, and Objective C. Although these languages may store information differently, the way they treat objects at the programming level is similar. The five primary concepts which distinguish object-oriented programming systems from other traditional systems are objects, methods, messages, classes, and inheritance.

Objects, Methods and Messages

Wegner defines an object as anything which has "a set of operations and a local shared state that remembers the effect of operations". They are the portions of a program that send and receive messages. (Wegner, 1989) Methods are the procedures or functions that are needed to control the object when it receives a message. A message corresponds to a function call in a traditional procedural language. In traditional programming systems, data and procedures are separate from each other. An object-oriented programming system combines the object and the procedures or methods needed to control the object. This system feature is called encapsulation. (Thomas, 1989). For example, in traditional programing, in order to draw a circle, square, or line, the program would have to contain separate procedures to draw each of those "objects":



An object-oriented program, on the other hand, would be organized in this manner:



The user tells the object what needs to be done and the object chooses the appropriate method from a table of operations the object knows how to do. In other words, instead of programs being organized into procedures that control global data, the data are bundled with the methods needed to access the data. Methods belonging to the object or data can only be triggered by sending the object a message.

Classes and Inheritance

The other major features specific to object-oriented programming systems are class and inheritance. A Class is a group of objects with common operations or consistent behavior (Shafer, 1988). Inheritance carries out code sharing by having the language rather than the programmer reuse code from one class to another related class. Inheritance, then, can be used as a vehicle for creating objects that share attributes with similar objects. (Thomas, 1989).

Wegner states that inheritance allows objects and their methods to begin not as new pieces of code; but as a single statement which refers to another class already defined within the program, and describes how the new object or class is different from the existing one (Wegner, 1989). This ability to inherit behavior results in what Shafer describes as " an ever-expanding and descending chain of heredity" (Shafer, 1988). Information is transmitted to all the places where it is needed. The end result would look something like the following example and would provide the ability to reuse program modules to both the system and the user:

Class: Transportation
  Class: Motor Vehicle
    Class: Sedan
      Instance: Full-size
      Instance: Mid-size
      Instance: Compact
    Class: Van
      Instance: Full
      Instance: Mini.

If the class Transportation has a method called "StartIt', every member of each subclass can use that method. If the message "StartIt" is sent to the object "mini", that object does not need to have a method called "StartIt". The system just passes the messages up the hierarchy until a corresponding method is found. All these features -- the encapsulation of objects and methods together, the use of messages to send and receive information, the creation of classes, and inheritance -- create an environment for a different and exciting style of programming.

Hypermedia and HyperTalk

The term hypertext was coined by Ted Nelson in the 1960's while working on a computer system called Xanadu. The Xanadu system was designed to use links to organize separate data banks of online information (Nelson, 1967). Hypertext, as defined by Nelson, denotes non-sequential reading and writing (Smith, 1988). Hypertext systems allow programmers and/or users to link information together. These links connect material in one document to related information in another. Hypermedia is used to describe a hypertext system which includes textual information plus additional information in other forms, such as different types of graphics, video, music and other sounds, and speech. With hypermedia, both the programmer and the user can create links to any or all of the information stored in the different formats (Por, 1987).

In hypertext systems information must be put into some sort of logical structure so it can be stored, found, and retrieved. Hypermedia systems parse information into small distinct modules, called nodes. Each node is made up of one concept or idea. The information included in a node is displayed on one computer screen at a time, as in Apple's HyperCard; or in groups of nodes, as in Xerox Palo Alto Research Center's (PARC) NoteCard system. (Fiderio, 1988). Links are used to connect the nodes by networking related text or information in other formats. A hypertext link is an electronic pointer that directs the user to associated information. Links are hypertext's transportation infrastructure; they are the routes the user follows to move from one node to another.

According to Fiderio, links have two basic characteristics: they must be traceable by the computer, and they must be able to route the user from one node to another as rapidly as possible. Links normally originate at a single point, which can be a single text character, a string, or an icon. These points identify links in a document. Guide, the hypertext package developed by OWL International for MS-DOS and Macintosh computers; and Apple's HyperCard for the Macintosh both refer to these points as buttons (Fiderio, 1988).

Comparison of OOPs and HyperTalk

There are some strong parallels between OOPs and HyperTalk. The fundamental concepts of OOPs (objects, messages, methods, classes, and inheritance) each have some counterparts in hypermedia systems. The degree to which the counterparts are intellectually and operationally parallel, however, differs markedly for particular hypermedia applications. In HyperTalk (the programming environment for HyperCard applications on Apple Macintosh), for example, cards, stacks, and buttons are analogous in some ways to objects; handlers and scripts are somewhat analogous to methods, and links are analogous to messages. HyperTalk, however, does not support inheritance.

Advantages

Both Hypermedia and OOPs have virtually unprotected, changeable, operating systems. The source code which specifies the configuration of the entire system is immediately available to any programmer, and that programmer's entire programming environment can be immediately changed in almost any manner just by changing a few lines of code. Both types of systems are

designed to emulate the way humans store and retrieve data. They therefore alleviate the frustration of retrieving data by allowing a more natural, nonlinear method of searching for information. They allow programming languages to be unusually small. The complex syntax needed to declare data types is unnecessary because everything is of only one type, an object. Its contribution is restricting the effects of change by placing a wall of code around each piece of data (object). All access to the data is handled by procedures that were put there to mediate access to the data, and allows for the reusability of code as well as easy maintenance or modification of code.

OOPs promises much more than a point-and-click window interface. Carefully designed class libraries let you quickly assemble applications from prefabricated parts. The power of even limited systems, like HyperCard and Lotus Macros, shows how quickly end users can build their own integrated applications, given the appropriate framework.

Disadvantages

The issue of standardization within object-oriented programming languages needs to be addressed. Establishing standards both in the source code and the terminology used would allow much more flexibility and portability of the source code from one context to another.

Hypertext is still in the developmental stages with a number of major problems still to be resolved. Even with extensive search capabilities, it's still not always possible to retrieve the desired information or avoid getting lost in the maze of hypertext links. There is also the difficulty of breaking a thought or a segment of information into a single node. Hypertext systems also have problems in helping the user visualize how the information is linked together; answering such questions as "What information is available?" and "How can I get to it?"

A major problem with hypertext, according to Fiderio, is that hypertext systems give the user control when the user actually needs guidance. With the user's ability to make his or her own links, comes the risk of making the wrong choice and getting lost in the maze of link paths created in the process.

The Future of OOPs and Hypertext

There is a great interest in the development of interactive learning systems. In their analysis of object-oriented programming, Dodani, et.al state that interactive software provides an important incentive to execute the user interface as an independent component. This separation, they claim, has several advantages. First, the user interface can be subdivided into components that can be joined in different combinations for specific purposes. Second, the interface can be modified quickly for use in other applications. Third, the interface can be changed or even removed without any harmful effects on the rest of the program. Finally, the interface can be developed on an ongoing basis, with prototypes continually being tested until the "perfect" design is developed (Dodani, Hughes, and Moshell, 1989).

In order to provide the type of separate system described above, we first have to change the organization of information into groups of concepts rather than isolated facts. We need to take advantage of the relationships between concepts; begin to consider what the learner understands; and to anticipate the learner's next choice, or to suggest a different direction (Weyer, 1988).

OOPs and hypermedia appear to be the likely tools not only for development of full programs; but also for the development of the interactive user interfaces to other, already developed programs. They are both systems which force the programmer to look at knowledge from the learner's perspective and needs. They provide the tools necessary for "navigating" through the maze of information resources available to the user. These systems hold the promise of allowing us to look at information the way we organize our own thoughts, and to make nonlinear connections between different bits of information.

Hypermedia and object-oriented programming systems are two very powerful tools, still in their developmental infancy. It will be exciting to see what the next five years bring in

the world of information search and retrieval systems.

Bibliography

Bush, V. (1945). "As We May Think." Atlantic Monthly, July, 1945, 101-8.

Dodani, M.; Hughes, C.; and Moshell, J. (1989) "Separation of Power." Byte, March, 1989, 255-262.

Fiderio, J., (1988). "A Grand Vision." Byte, October, 1988, 237-244.

Nelson, T. (1967). "Getting it out of our system." In Information retrieval: A critical review, ed. G. Schechter, Washington, DC: Thompson Books.

Por, G., (1987). "Hypermedia and Higher Education." Computer Currents, August, 1987, 14-16.

Shafer, D., (1988). HyperTalk Programming, Indianapolis, Indiana: Hayden Books, 1988.

Smith, K., (1988). "Hypertext -- Linking to the Future." Online, March, 1988, 32 -40.

Thomas, D. (1989). "What's In An Object?" Byte, March, 1989,231-240.

Wegner, P. (1989). "Learning The Language" Byte, March, 1989, 245-253.

Weyer, S. (1988). "As We May Learn." In Interactive Multimedia:Visions of Multimedia for Developers, Educators, and Information Providers, ed. S. Ambron and K. Hooper, Redmond, Washington: Microsoft Press, 1988.

Lynn K. Milet
Francis A. Harvey
Lehigh University
Media Center, Fairchild-Martindale Library
Bethlehem, PA 18015
(215) 758-3059